Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 3, Juli 2023
DOI : https://doi.org/10.33395/sinkron.v8i3.12423
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Zero Knowledge Proof for SNAP (Standar Nasional OPEN API Pembayaran) in Indonesia

Moehammad Ramadhoni[1)], Handri Santoso[2)*]
[1,2)]Universitas Pradita, Indonesia,
[1)]moehammad.ramadhoni@student.pradita.ac.id, [2)]handri.santoso@pradita.ac.id

Submitted : May 13, 2023 | **Accepted** : Jun 7, 2023 | **Published** : Jul 1, 2023

Abstract: SNAP (Standar Nasional OPEN API Pembayaran) is an implementation of open banking for encouraging digital transformation in the banking industry. SNAP was submitted by several sub-working groups formed jointly by ASPI and the Bank of Indonesia. In the document Pedoman Tata Kelola (Bank of Indonesia, n.d.), a customer data protection mechanism exists between the bank, the owner of Open API, and the user of Open API. However, there is no data protection process carried out by consumers so third parties, that use the Open API of the bank, do not need to know the customer's data. Based on the web3 with zero-knowledge proofs protocol, users can store data and transmit only in encrypted form which can only be opened by calculating the data with a pre-agreed smart contract. Banks can work like a decentralized network on web3, where the process of calculating proof and witness is carried out by the bank. Proof and witness are calculated using a zero-knowledge proof protocol, making duplicating difficult. For this reason, we propose a new architecture using smart contracts between banks and customers using the ZK-SNARK method using gnark library in the Golang programming language. Therefore, there is no significant performance difference between using ZK-SNARK and without ZK-SNARK in the API call process.

Keywords: data protection, gnark, SNAP, zero-knowledge proofs, ZK-SNARK

INTRODUCTION

On 16 August 2021, Bank Indonesia verified a document regarding the use of an Open API for payments in Indonesia named SNAP (Standar Nasional OPEN API Pembayaran). This document was initiated by many parties who are members of a sub-working group called ASPI (Asosiasi Sistem Pembayaran Indonesia). The document already contains a chapter on consumer data protection and protection mechanisms in the event of a data leak.

In addition, there are Standar Data Spesifikasi Teknis SNAP (Bank of Indonesia, n.d.) and Standar Teknis Keamanan (Bank of Indonesia, n.d.) documents that have been verified by Bank Indonesia and ASPI which discuss technically how to secure data and the API calling process, such as using public and private keys, using encoding authentication in each header, and the use of signatures in headers. The document also contains the prerequisites needed so that banks and third parties can use the Open

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 3, Juli 2023 e-ISSN : 2541-2019
DOI : https://doi.org/10.33395/sinkron.v8i3.12423 p-ISSN : 2541-044X

API, such as the availability of written policies, fulfillment of certification, and availability of tools to detect fraud.

This is a commitment from ASPI members and the Bank of Indonesia to secure and protect consumer data from irresponsible parties. In the future, it is hoped that other industries will participate in creating standard-compliant Open APIs as well so that consumers have convenience in the digital era without having to sacrifice data security.

However, consumers should be included in the data security process. Consumers should also be aware that data security is a crucial part of the digital era, and will increase data security from these consumers so that only banks and consumers know consumer data. Third parties do not need to acknowledge user data, such as account numbers, user account names, and bank branches.

Therefore, we propose a payment system architecture using Zero Knowledge Proof (hereinafter referred to as ZKP) using Cryptography Smart Contracts between consumers and related banks. Consumers do not need to send personal data to third parties, such as account numbers, usernames, and card numbers. Therefore, banks still can verify consumers using proof that sent by consumers.

The ZKP technique used in this journal is ZK-SNARK (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge). The ZK-SNARK method is non-interactive which the *prover* and *verifier* agree to use the *shared key* as a way to generate and verify *proof*. All operations are performed in one call between the *prover* and *verifier*.

One of the important processes in ZK-SNARK is the process of generating publicly known parameters (generally known as Common Reference String(CRS)) because this is very important in the security of the protocol. If the entropy (randomness) used to generate CRS is known to an irresponsible third party, it is feared that the party can produce *fake proof*.

LITERATURE REVIEW

Goldwasser, Micali, and Rackoff (Golwasser et al., 1985) introduced a zero-knowledge proof that allows a prover to prove to a verifier that he knows information without having to reveal what the information is. The basic concept of ZKP is that the prover could exchanges messages with the verifier, where the prover tries to convince the verifier that the prover knows some information without having to tell the verifier the information.

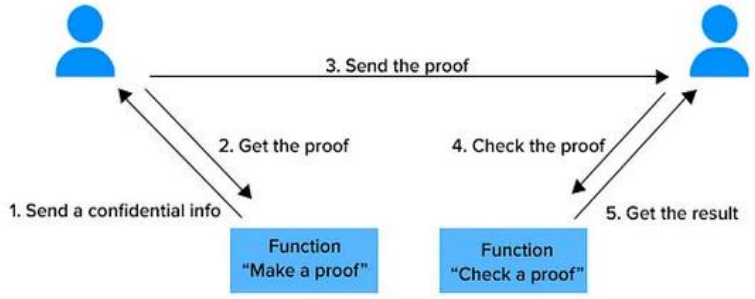

Fig 1: Zero-knowledge proofs
Source : https://appinventiv.com/blog/zero-knowledge-proof-blockchain/

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 3, Juli 2023 e-ISSN : 2541-2019
DOI : https://doi.org/10.33395/sinkron.v8i3.12423 p-ISSN : 2541-044X

Zero-knowledge proofs represented a breakthrough in applied cryptography, as they promised to improve security of information for individuals. Consider how you might prove a claim about nationality. To prove that, we have to provide an evidence to support our answer, for example, a KTP or driver's license. With identity theft becoming a critical issue, there are calls for more privacy-protecting means of sharing sensitive information.

Zero-knowledge proofs solve this problem by eliminating the need to reveal information to prove validity of claims (Ethereum.Org, n.d.). The zero-knowledge protocol uses the statement (called a 'witness') as input to generate a succinct proof of its validity. This proof provides strong guarantees that a statement is true without exposing the information used in creating it.

A zero-knowledge protocol must satisfy the following criteria, first **completeness,** which means if the input is valid, the zero-knowledge protocol always returns 'true', then is **soundness,** which means if the input is invalid, it is theoretically impossible to fool the zero-knowledge protocol to return 'true', and the last **zero-knowledge**, means the verifier learns nothing about a statement beyond its validity or falsity (they have "zero knowledge" of the statement).

While revolutionary, interactive proving had limited usefulness since it required the two parties to be available and interact repeatedly. To solve this problem, Manuel Blum, Paul Feldman, and Silvio Micali (Blum et al. 1988) suggested the first non-interactive zero-knowledge proofs where the prover and verifier have a shared key. This allows the prover to demonstrate their knowledge of some information (i.e., witness) without providing the information itself.

Unlike interactive proofs, non-interactive proofs required only one round of communication between participants (prover and verifier). The prover passes the secret information to a special algorithm to compute a zero-knowledge proof. This proof is sent to the verifier, who checks that the prover knows the secret information using another algorithm.

Kilian (Kilian, 1992) gave the first sublinear communication zero-knowledge argument that sends fewer bits than the size of the statement to be proved. Micali (Micali, 2000) proposed sublinear size arguments by letting the prover in a communication efficient argument compute the verifier's challenges using a cryptographic function, and as remarked in Kilian (Kilian, 1992) this leads to sublinear size NIZK proofs when the interactive argument is public coin.

Groth, Ostrovsky and Sahai (Groth et al., 2012) introduced pairing-based NIZK proofs, yielding the first linear size proofs based on standard assumptions. Lipmaa (Lipmaa, 2012) used an alternative construction based on progression-free sets to reduce the size of the Common Reference String (CRS). Groth (Groth, 2016) added a preprocessing scheme to the circuit so that the argument size is constant and the verification time is faster than before.

These schemes above are used in the ZK-SNARK (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) protocol. ZK-SNARK protocol has the following qualities, **zero-knowledge**, which means a verifier can validate the integrity of a statement without knowing anything else about the statement. The only knowledge the verifier has of the statement is whether it is true or false, then **succinct**, which means The zero-knowledge proof is smaller than the witness and can be verified quickly, also **non-interactive**, which means the prover and verifier only interact once, on forth **argument**, which means he proof satisfies the 'soundness' requirement, so cheating is extremely unlikely, and the last is **(of) knowledge,** which means the zero-knowledge proof cannot be constructed without access to the secret information (witness). It is difficult, if not impossible, for a prover who doesn't have the witness to compute a valid zero-knowledge proof.

This protocol makes ZK-SNARK a good choice for authentication applications (Ethereum.Org, n.d.), especially for payments using the public Open API. ZK-SNARK ensures that authentication is very simple, with only one connection, and the size of witness that is sent is small and constant so that the calculation time needed to verify is very small.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 3, Juli 2023
DOI : https://doi.org/10.33395/sinkron.v8i3.12423
e-ISSN : 2541-2019
p-ISSN : 2541-044X

METHOD

Based on the SNAP document from the Bank of Indonesia, the Open API communication flow is consumers, third parties, then banks as API owners. The consumer enters personal data on a third-party website which is then forwarded by the third party to the bank. In the process, third parties can store important data and then use it in other processes. Therefore, consumers should not provide data to third parties, but send data that has been encrypted and can only be verified by the bank.

To meet security requirements on the consumer side, consumers must encrypt data before entering it into third-party applications. Data encryption is used on the web3 platform for public coins implementation. For this reason, the data encryption process on web3 adapted to increase connection security between customers, third parties, and banks. The protocol used in the encryption process is ZK-SNARK because the non-interactive process that does not require many connections is very suitable for the network architecture in SNAP.

When compared to the concepts used on the blockchain in public coins, consumers store all data in the form of smart contracts. Then when a transaction occurs, the consumer will send special public data where to solve the transaction certain calculations are required based on the smart contract owned by the consumer in the network. This journal proposes a zero-knowledge proofs method similar to the public coin architecture with slight changes to the proof calculations to adapt to the architecture currently used by banks in Indonesia.

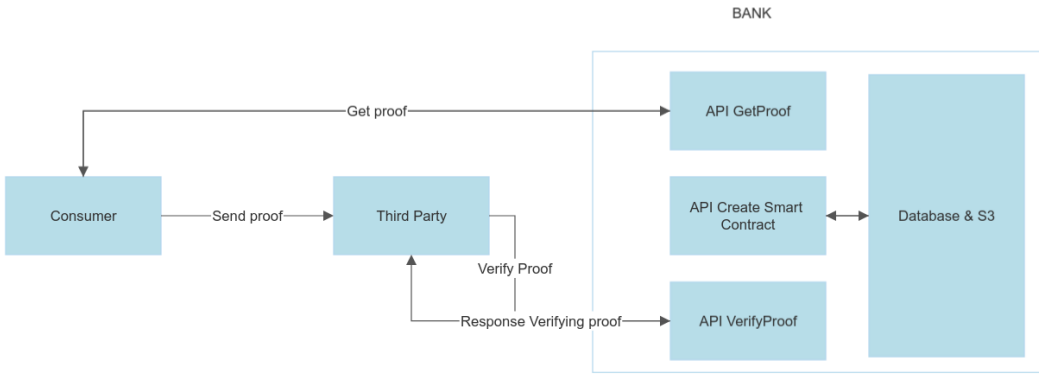

Fig 2: Proposed architecture

The implementation uses the gnark library and the Golang programming language. From the picture above, it is known that the implementation created three APIs as simulations, namely the Ping API, GetProof API, and VerifyProof API. API Ping is a simulation of a connection between a third party and a bank without using ZKP verification. The getProof API is an API used by consumers to ask banks to calculate proofs based on pre-made smart contracts. Finally, the verifyProof API is an API used to verify proofs sent by third parties to banks.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 3, Juli 2023
DOI : https://doi.org/10.33395/sinkron.v8i3.12423
e-ISSN : 2541-2019
p-ISSN : 2541-044X

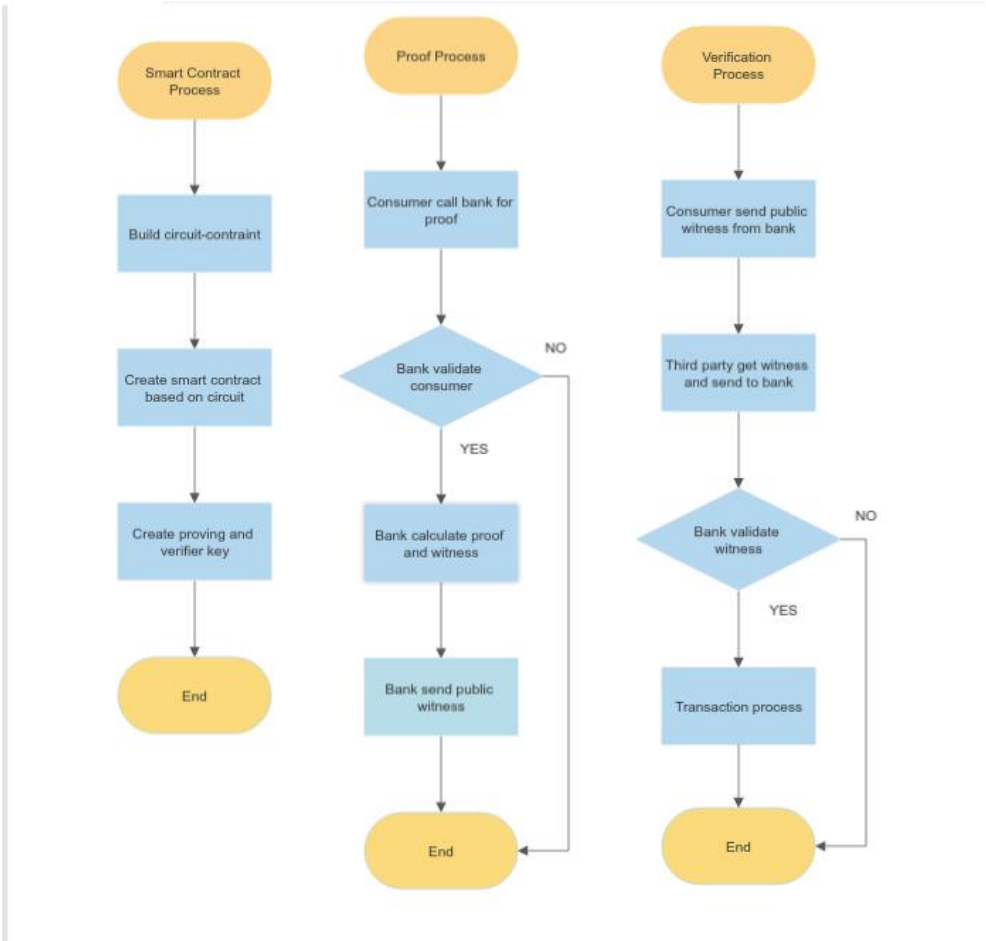

Fig 3: Flowchart proposed architecture`

After the proposed design is implemented, the code needs testing to find out the significance of the difference between our novel architecture and existing architecture. For this, the built-in tools from Golang, namely benchmarks, are used. In this case, the performance of the three APIs will be calculated within 1 second. Apart from that, benchmark results will also be calculated from three different circuit constraints to find out how many processes occur and the resources required in 1 second of the ZK-SNARK process.

RESULT

Benchmarks are used to get the performance level of a code segment when it is called many times and compared to other functions with the same standard. Golang has several built-in tools that usually use for benchmarking with accurate results. To get good results, we must isolate the machine used for benchmarking and each benchmark must have the same environment so that stable results will be obtained for each function being measured.

```
❯ go test -bench=. ./app/handlers/web
2023/05/10 02:19:08 [INFO] Connected to POSTGRES TYPE = 127.0.0.1 | LogMode = false
goos: linux
goarch: amd64
pkg: smart-contract-service/app/handlers/web
cpu: 11th Gen Intel(R) Core(TM) i5-113567 @ 2.40GHz
BenchmarkHTTP_PingHandler/Endpoint:_GET_/ping-8          2559588          820.0 ns/op          394 B/op          2 allocs/op
BenchmarkHTTP_GetProof/Endpoint:_GET_/proof-8               8718       121492 ns/op        15978 B/op        288 allocs/op
BenchmarkHTTP_VerifyProof/Endpoint:_POST_/proof-8        2259682          542.7 ns/op          290 B/op          4 allocs/op
PASS
ok      smart-contract-service/app/handlers/web 5.445s
```

Fig 4: Benchmark code result

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 3, Juli 2023
DOI : https://doi.org/10.33395/sinkron.v8i3.12423
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Table 1: Benchmark of ZKP function

Function	Number of Processes	Time per Process	Byte per Process	Memory per Process
Ping	2.559.588	820 ns/op	394 B/op	2 allocs/op
Get Proof	8.718	121.492 ns/op	15.978 B/op	208 allocs/op
Verify Proof	2.259.682	542,7 ns/op	298 B/op	4 allocs/op

The table above describes the allocation of time, bytes, and memory for each function that is executed within 1 second. Consider the table above, it is found that the time difference between using ZK-SNARK and without ZK-SNARK is not significant. The difference between the Ping function and verify proof is insignificant, the biggest difference is in the get proof function, which is the function used to calculate public witnesses that consumers use to verify third-party applications. This means authentication using ZK-SNARK can be implemented in SNAP governance to add consumer data security features.

Table 2: ZK-SNARK difference based on circuit-constraint

Circuit	Constraint	Number of Processes	Time per Process	Byte per Process	Memory per Process
Elliptic Curve	3	926	1,2 ms/op	79 KB/op	353 allocs/op
Hash	331	40	30 ms/op	1.8 MB/op	7.475 allocs/op
edDSA	6497	3	467 ms/op	32 MB/op	203.210 allocs/op

The table above illustrates the ZK-SNARK process from the start of creating a smart contract to the verification process in 1 second. From the table above can be seen that the calculation process by adding data constraints, both public and secret, requires processing time which will increase along with the number of constraints. However, the addition of time and memory allocation does not have constant straight line because it uses groth16 (Groth, 2016) calculations where the proof calculation process made smaller than the witness so that the verification process will take less time than when calculating smart contracts and proofs.

DISCUSSIONS

From the benchmark results, we can see that authentication ZK-SNARK using the Golang programming language can be implemented and does not interfere with transferring data from consumers, to third parties, and to banks. Authentication even improves the digitization process without harming consumers due to the fact that sensitive data can be accessed by many parties.

The get proof function needs more time to process than the verify proof function, which means ZK-SNARK need more resource when creating proof and witness. However, it only needs 542,7 ns/op for verifying the function. This happens because, in Web3, verification should be done immediately after proofs are introduced, so miners can mine the blockchain and solve its proof-of-work.

Also based on Table 2, we see that the edDSA algorithm has the largest constraint that makes ZK-SNARK need more time to process than other algorithms. It needs 467 ms/op for creating a smart

*name of corresponding author

contract until verifying its transaction. So, we proposed bank to make the smart contract and create proofs for customers because they have great resources and could do this algorithm more efficiently.

Based on two tables above, the ZK-SNARK algorithm has shown not affect function performance and significantly increase response time. The smart contracts contained in the ZKP process using ZK-SNARK are also easy to implement and can use various circuit constrains depending on the level of security desired by the user and the bank concerned.

CONCLUSION

From the research results above, several conclusions were obtained. First conclusion is ZKP with the ZK-SNARK protocol using the gnark library in the Golang programming language does not significantly affect the response time and performance of the function called. Then, ZKP can increase the security of user data following the Pedoman Tata Kelola SNAP documentation, so it is recommended to use it in the next implementation update. Finally, further research is needed to process transactions using ZKP that require two or more parties (multi-party signs).

REFERENCES

Bank of Indonesia (n.d) Pedoman Tata Kelola SNAP. Retrieved May 01, 2023, from https://bi.go.id/id/layanan/Standar/SNAP/Documents/SNAP_Pedoman_Tata_Kelola.pdf

Bank of Indonesia (n.d) Standar Data Spesifikasi Teknis SNAP. Retrieved May 01, 2023, from https://apidevportal.bi.go.id/snap/docs/standar-data-spesifikasi-teknis

Bank of Indonesia (n.d) Standar Teknis Keamanan SNAP. Retrieved May 01, 2023, from https://apidevportal.bi.go.id/snap/docs/standar-teknis-keamanan

Barreto, P. S. L. M., & Naehrig, M. (2006). Pairing-friendly elliptic curves of prime order. In B. Preneel & S. Tavares (Eds.), *Selected Areas in Cryptography* (Vol. 3897, pp. 319–331). Springer Berlin Heidelberg. https://doi.org/10.1007/11693383_22

Bin Uzayr, S. (2022a). *Mastering golang: A beginner's guide* (1st ed.). CRC Press. https://doi.org/10.1201/9781003310457

Bin Uzayr, S. (2022b). *Golang: The ultimate guide* (1st ed.). CRC Press. https://doi.org/10.1201/9781003309055

Blum, M., Feldman, P., & Micali, S. (1988). Non-interactive zero-knowledge and its applications. *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing - STOC '88*, 103–112. https://doi.org/10.1145/62212.62222

Buterik, Vitalik. (2021) *An approximate introduction to how zk-SNARKs are possible*. Retrieved May 06, 2023, from https://vitalik.ca/general/2021/01/26/snarks.html

Dwivedi, A. D., Singh, R., Ghosh, U., Mukkamala, R. R., Tolba, A., & Said, O. (2022). Privacy preserving authentication system based on non-interactive zero knowledge proof suitable for Internet of Things. *Journal of Ambient Intelligence and Humanized Computing*, *13*(10), 4639–4649. https://doi.org/10.1007/s12652-021-03459-4

Dymora, P., & Paszkiewicz, A. (2020). Performance analysis of selected programming languages in the context of supporting decision-making processes for industry 4.0. *Applied Sciences (Switzerland)*, *10*(23), 1–17. https://doi.org/10.3390/app10238521

Effendy, F., Taufik, & Adhilaksono, B. (2019). Performance Comparison of Web Backend and Database: A Case Study of Node.JS, Golang and MySQL, Mongo DB. *Recent Advances in Computer Science and Communications*, *14*(6), 1955–1961. https://doi.org/10.2174/2666255813666191219104133

El Housni, Y., & Guillevic, A. (2022). Families of snark-friendly 2-chains of elliptic curves. In O. Dunkelman & S. Dziembowski (Eds.), *Advances in Cryptology – EUROCRYPT 2022* (Vol. 13276, pp. 367–396). Springer International Publishing. https://doi.org/10.1007/978-3-031-07085-3_13

Ethereum.Org .(n.d). *Zero-knowledge proofs*. Retrieved May 06, 2023, from https://ethereum.org

Gaba, G. S., Hedabou, M., Kumar, P., Braeken, A., Liyanage, M., & Alazab, M. (2022). Zero knowledge proofs based authenticated key agreement protocol for sustainable healthcare. *Sustainable Cities and Society*, *80*, 103766. https://doi.org/10.1016/j.scs.2022.103766

*name of corresponding author

Goldwasser, S., Micali, S., & Rackoff, C. (1985). The knowledge complexity of interactive proof-systems. *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing - STOC '85*, 291–304. https://doi.org/10.1145/22145.22178

Gong, Y., Jin, Y., Li, Y., Liu, Z., & Zhu, Z. (2022). Analysis and comparison of the main zero-knowledge proof scheme. In *Proceedings - 2022 International Conference on Big Data, Information and Computer Network, BDICN 2022* (pp. 366–372). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/BDICN55575.2022.00074

Groth, J. (2006). Simulation-sound nizk proofs for a practical language and constant size group signatures. In X. Lai & K. Chen (Eds.), *Advances in Cryptology – ASIACRYPT 2006* (Vol. 4284, pp. 444–459). Springer Berlin Heidelberg. https://doi.org/10.1007/11935230_29

Groth, J. (2009). Linear algebra with sub-linear zero-knowledge arguments. In S. Halevi (Ed.), *Advances in Cryptology—CRYPTO 2009* (Vol. 5677, pp. 192–208). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-03356-8_12

Groth, J. (2016). On the size of pairing-based non-interactive arguments. In M. Fischlin & J.-S. Coron (Eds.), *Advances in Cryptology – EUROCRYPT 2016* (Vol. 9666, pp. 305–326). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-49896-5_11

Groth, J., Ostrovsky, R., & Sahai, A. (2012). New techniques for noninteractive zero-knowledge. *Journal of the ACM*, *59*(3), 1–35. https://doi.org/10.1145/2220357.2220358

Groth, J., & Sahai, A. (2012). Efficient noninteractive proof systems for bilinear groups. *SIAM Journal on Computing*, *41*(5), 1193–1232. https://doi.org/10.1137/080725386

Harjoseputro, Y., Albertus Ari Kristanto, & Joseph Eric Samodra. (2020). Golang and NSG Implementation in REST API Based Third-Party Sandbox System. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *4*(4), 745–750. https://doi.org/10.29207/resti.v4i4.2218

Hunacek, M. (2023). *Introduction to number theory* (1st ed.). Chapman and Hall/CRC. https://doi.org/10.1201/9781003318712

Kilian, J. (1992). A note on efficient zero-knowledge proofs and arguments (Extended abstract). *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing - STOC '92*, 723–732. https://doi.org/10.1145/129712.129782

Kilian, J. (1995). Improved efficient arguments. In D. Coppersmith (Ed.), *Advances in Cryptology—CRYPTO' 95* (Vol. 963, pp. 311–324). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-44750-4_25

Kim, J., Lee, J., & Oh, H. (2020). Simulation-extractable zk-SNARK with a single verification. *IEEE Access*, *8*, 156569–156581. https://doi.org/10.1109/ACCESS.2020.3019980

Konkin, A., & Zapechnikov, S. (2023). Zero knowledge proof and ZK-SNARK for private blockchains. *Journal of Computer Virology and Hacking Techniques*. https://doi.org/10.1007/s11416-023-00466-1

Kristanto, A. A., Harjoseputro, Y., & Samodra, J. E. (2020). Implementasi Golang dan New Simple Queue pada Sistem Sandbox Pihak Ketiga Berbasis REST API. *Jurnal Rekayasa Sistem Dan Teknologi Informasi (RESTI)*, *4*(4), 745–750.

Li, W. H., Zhang, Z. Y., Zhou, Z. B., & Deng, Y. (2022, July 1). An Overview on Succinct Non-interactive Zero-knowledge Proofs. *Journal of Cryptologic Research*. Chinese Association for Cryptologic Research. https://doi.org/10.13868/j.cnki.jcr.000525

Lipmaa, H. (2012). Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In R. Cramer (Ed.), *Theory of Cryptography* (Vol. 7194, pp. 169–189). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-28914-9_10

Micali, S. (2000). Computationally sound proofs. *SIAM Journal on Computing*, *30*(4), 1253–1298. https://doi.org/10.1137/S0097539795284959

Setty, S. (2020). Spartan: Efficient and general-purpose zksnarks without trusted setup. In D. Micciancio & T. Ristenpart (Eds.), *Advances in Cryptology – CRYPTO 2020* (Vol. 12172, pp. 704–737). Springer International Publishing. https://doi.org/10.1007/978-3-030-56877-1_25

Toyib, R., & Darnita, Y. (2020). Pengamanan Data Teks Dengan Menggunakan Algoritma Zero-Knowledge Proof. *JURNAL MEDIA INFOTAMA*, *16*(1). https://doi.org/10.37676/jmi.v16i1.1114

*name of corresponding author

Tyagi, S., & Kathuria, M. (2022). Role of Zero-Knowledge Proof in Blockchain Security. In *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing, COM-IT-CON 2022* (pp. 738–743). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/COM-IT-CON54601.2022.9850714

Ullah, S., Zheng, J., Din, N., Hussain, M. T., Ullah, F., & Yousaf, M. (2023). Elliptic Curve Cryptography; Applications, challenges, recent advances, and future trends: A comprehensive survey. *Computer Science Review*, *47*, 100530. https://doi.org/10.1016/j.cosrev.2022.100530

Wahby, R. S., Tzialla, I., Shelat, A., Thaler, J., & Walfish, M. (2018). Doubly-efficient zksnarks without trusted setup. *2018 IEEE Symposium on Security and Privacy (SP)*, 926–943. https://doi.org/10.1109/SP.2018.00060

*name of corresponding author